Conserving Fuel in Statistical Language Learning: Predicting Data Requirements *

Mark Lauer
65 Epping Road, North Ryde NSW 2113, Australia
Email: t-markl@microsoft.com

Abstract

The paradigm for NLP known as STATISTICAL LANGUAGE LEARNING (SLL) has flourished in recent times, being seen as a quick and easy way to get off the ground. Research systems have been launched at many NLP problems including sense disambiguation (Yarowsky, 1992), anaphora resolution (Dagan and Itai, 1990), prepositional phrase attachment (Hindle and Rooth, 1993) and lexical acquisition (Brent, 1993). This has all been fueled by the large text corpora which are increasingly available (Marcus *et al.*, 1993). Since these systems learn to navigate language by consuming text, they are critically dependent on the data that drives them.

In this paper I address the practical concern of predicting how much training data is sufficient for a given system. First, I briefly review earlier results and show how these can be combined to bound the expected accuracy of a mode-based learner as a function of the volume of training data. I then develop a more accurate estimate of the expected accuracy function under the assumption that inputs are uniformly distributed. Since this estimate is expensive to compute, I also give a close but cheaply computable approximation to it. Finally, I report on a series of simulations exploring the effects of inputs that are not uniformly distributed.

1 Background

1.1 Do We Need To Know?

Even though text is becoming increasingly available, it is often expensive, especially if it must be annotated. Consider the decisions facing the SLL technology consumer, that is, the architect of a planned commercial NLP system. For each module which is to employ SLL, an appropriate technique must be selected. If different techniques require different amounts of data to achieve a given accuracy, the architect would like to know what these requirements are in advance in order to make an informed choice.

Further, once the technique is chosen, she must decide how much data to collect or purchase for training. Because this data can be expensive, foreknowledge of data requirements is highly valuable. Thus, in order to make statistical NLP technology practical, a predictive theory of data requirements is needed. Despite this need, very little attention has been paid to the problem.[1]

*This paper has been accepted for publication at the Eigth Australian Joint Conference on Artificial Intelligence, Canberra, 1995.

[1]See de Haan (1992) for an investigation of sample sizes for linguistic studies.

1.2 Foundations For A Theory

All the SLL systems mentioned above employ knowledge gained from a corpus to make decisions. Abstractly, this knowledge can be represented as a mapping from observable features (inputs) to decision outcomes (outputs). Following Lauer (1995) I will call each distinguished input a BIN and each possible output a VALUE. There is a probability distribution across the bins representing how instances fall into bins. Also, for each bin, there is a probability distribution across the set of values representing how instances in that bin take on values. For the system to perform accurately, most (but not necessarily all) of the instances falling in a particular bin must have the same value.

In what follows I will make several assumptions: Training and test data are drawn from the same distributions. The set of possible values is binary (examples include Hindle and Rooth, 1993 and Lauer, 1994). The probability of the most likely value in each bin is constant.[2] Finally, I will only consider a simple learning algorithm: collect the training instances falling into each bin and then select the most frequent value for each. This mode-based learner is employed directly in the unigram tagger of Charniak (1993, p49) and is at the heart of many systems.

1.3 Optimal Accuracy

There are two sources of error in statistical language learners of the kind we are considering. First, since the values are not necessarily fully determined by the bins, no matter what value the learner assigns to a bin there will always be errors (the optimal error rate). Second, since training data is limited, the learner may not have sufficient data available to acquire accurate rules. The combination of these sources of error results in some degree of inaccuracy for the system. We are interested in estimating the accuracy for various volumes of training data. Since the optimal error rate is independent of the amount of training data, it will always exist no matter how much data is used. As the amount of training data increases we expect the accuracy to get closer to this optimal.

Let B be the set of bins, V the set of values, $\Pr(b)$ the probability that an instance falls into the bin b and $\Pr(v \mid b)$ the probability of the value v given the bin b. If we denote the most likely value in each bin as $v_b = \text{argmax}_{v \in V} \Pr(v \mid b)$, then the expected value of the optimal accuracy is determined by the likelihood of this value occurring in each bin.

$$\text{OA} = \sum_{b \in B} \Pr(b) \Pr(v_b \mid b) \tag{1}$$

If we know the probability that an algorithm will learn the value v for the bin b (denote this $\Pr(\text{learn}(b) = v)$), then we can also calculate the expected accuracy rate:

$$\text{EA} = \sum_{b \in B} \Pr(b) \sum_{v \in V} \Pr(\text{learn}(b) = v) \Pr(v \mid b) \tag{2}$$

In Lauer (1995) several results are shown concerning the relationship of these two values. I will summarise these in section 2.1 (see equations (3) and (4)).

[2] Note that this does not require that the most likely value be the same value in each bin; only that whatever the most likely value is has a constant probability.

2 Existing Work

2.1 Empty Bins and Non-empty Bins

The most severe result of insufficient training data is that some bins can go without any training instances. Since the learner has no indications about likely values for the bin it will be forced to guess. To estimate how often this will occur, consider the way in which m training instances would fall into the bins. For each bin, the probability that no training instances fall into it is:

$$\Pr(\text{count}(b) = 0) = (1 - \Pr(b))^m$$

I will call such bins EMPTY BINS.

In Lauer (1995) it is shown that for any bin b:

$$\Pr(\text{count}(b) = 0) < e^{-m/|B|} \tag{3}$$

Lauer (1995) also bounds the expected accuracy of the mode-based learner when all bins are guaranteed to have at least one training instance. When this is the case, it is shown that the expected error rate is always no worse than twice the optimal error rate.

$$\text{EA} \geq (1 - 2(1 - \text{OA})) \tag{4}$$

This is quite a useful result, since we expect the optimal accuracy to be fairly high. If the optimal predictions are 90% accurate, then a mode-based learner will be at least 80% accurate after learning on just one instance per bin.

2.2 Overall Expected Accuracy

Unfortunately, we cannot normally guarantee that no bins will be empty, since the corpus is typically a random sample. However, we can combine equations (3) and (4) to arrive at a bound for the overall expected accuracy after training on a random sample. Over non-empty bins, we know that the error rate is no worse than twice the optimal error rate for those bins. Since we have assumed that $\Pr(v_b \mid b)$ is constant (call this p), we can infer that the optimal accuracy for the non-empty bins is the same as the optimal accuracy on all bins. Thus:

$$\begin{aligned}
\text{EA} &= \Pr(\text{non-empty})\text{EA}(\text{non-empty}) + \Pr(\text{empty})\text{EA}(\text{empty}) \\
&\geq (1 - e^{-m/|B|})\text{EA}(\text{non-empty}) + (e^{-m/|B|})\text{EA}(\text{empty}) \\
&\geq (1 - e^{-m/|B|})(1 - 2(1 - \text{OA})) + \frac{1}{2}e^{-m/|B|} \\
&= (1 - e^{-m/|B|})(2p - 1) + \frac{1}{2}e^{-m/|B|}
\end{aligned} \tag{5}$$

The second step follows from the fact that EA(non-empty) $\geq$ EA(empty) and equation (3). The third step follows from equation (4).

3 Theory

3.1 Estimating Expected Accuracy

Given the assumptions in section 1.2, we can arrive at a better estimate of the expected accuracy when the distribution of bins is uniform (that is, $\Pr(b) = \frac{1}{|B|}$). Let the total number

of training instances in a bin b be n and the number of these instances with value v be $\text{count}(v, b)$:

$$\Pr(\text{count}(v,b) > n/2) = \sum_{i=\lceil n+1/2 \rceil}^{n} \binom{n}{i} \Pr(v \mid b)^i (1 - \Pr(v \mid b))^{n-i}$$

If n is even, we must also add an additional term of $1/2\binom{n}{n/2}\Pr(v \mid b)^{n/2}(1 - \Pr(v \mid b))^{n/2}$. This is because when there are equal numbers of both values in the bin, a random guess yields an expected accuracy of 50%. In the arguments below, I will treat all values of n as odd in order to simplify. The reader may check for herself that the results hold generally when the above extra term is included.

Using the fact that V is binary, the total expected accuracy for test instances in bin b when it contains n training instances is:

$$\Pr(v = \text{argmax}_{v' \in V} \text{count}(v', b)) = \sum_{v \in V} \Pr(v \mid b) \sum_{i=(n+1)/2}^{n} \binom{n}{i} \Pr(v \mid b)^i (1 - \Pr(v \mid b))^{n-i}$$

By summing over all possible numbers of training instances in a bin, we can arrive at an expression for the expected accuracy across all bins as follows:

$$\text{EA} = \sum_{b \in B} \Pr(b) \sum_{n=0}^{m} \text{binomial}(n; m, \Pr(b)) \sum_{v \in V} \Pr(v \mid b) \sum_{i=(n+1)/2}^{n} \text{binomial}(i; n, \Pr(v \mid b))$$

where $\text{binomial}(j; k, p) = \binom{k}{j} p^j (1-p)^{k-j}$.

To simplify this I have defined a function as follows:

$$G(m, r, p) = \sum_{n=0}^{m} \text{binomial}(n; m, r) \sum_{i=(n+1)/2}^{n} \text{binomial}(i; n, p)$$

A result which may be easily obtained by expansion is:

$$G(m, r, 1-p) = 1 - G(m, r, p) \tag{6}$$

Using the assumptions in section 1.2 and the uniform bin probabilities we can now proceed to simplify:

$$\begin{aligned}
\text{EA} &= \sum_{b \in B} \frac{1}{\mid B \mid} \sum_{n=0}^{m} \text{binomial}(n; m, \frac{1}{\mid B \mid}) \sum_{v \in V} \Pr(v \mid b) \sum_{i=(n+1)/2}^{n} \text{binomial}(i; n, \Pr(v \mid b)) \\
&= \sum_{b \in B} \frac{1}{\mid B \mid} \sum_{v \in V} \Pr(v \mid b) \sum_{n=0}^{m} \text{binomial}(n; m, \frac{1}{\mid B \mid}) \sum_{i=(n+1)/2}^{n} \text{binomial}(i; n, \Pr(v \mid b)) \\
&= \sum_{b \in B} \frac{1}{\mid B \mid} \sum_{v \in V} \Pr(v \mid b) G(m, \frac{1}{\mid B \mid}, \Pr(v \mid b)) \\
&= \sum_{b \in B} \frac{1}{\mid B \mid} (pG(m, \frac{1}{\mid B \mid}, p) + (1-p)G(m, \frac{1}{\mid B \mid}, 1-p)) \\
&= (1 - p + (2p-1)G(m, \frac{1}{\mid B \mid}, p)) \qquad (7)
\end{aligned}$$

The last step uses equation (6) and $\sum_{b \in B} \frac{1}{|B|} = 1$.

3.2 A Computable Bound for G

The main difficulty with the function G is the appearance of $\binom{m}{n}$. Most corpus-based language learners use large corpora, so we expect the number of training instances, m, to be very large. So we need a more easily computable version of G. The following argument leads to a fairly tight lower bound to G for suitably chosen values of k_j (see below):

$$\begin{aligned}
G(m,r,p) &= \sum_{n=0}^{m} \sum_{i=(n+1)/2}^{n} \text{binomial}(n;m,r)\text{binomial}(i;n,p) \\
&= \sum_{j=0}^{(m-1)/2} \sum_{n=2j+1}^{m} \text{binomial}(n;m,r)\text{binomial}(n-j;n,p) \\
&= \sum_{j=0}^{(m-1)/2} \sum_{n=2j+1}^{m} \binom{m}{n} r^n (1-r)^{m-n} \binom{n}{j} p^{n-j}(1-p)^j \\
&= \sum_{j=0}^{(m-1)/2} (1-r)^m (\frac{1-p}{p})^j \sum_{n=2j+1}^{m} \frac{m!}{(m-n)!}(1-r)^{-n}\frac{p^n}{n!}\binom{n}{j} r^n \\
&\geq \sum_{j=0}^{(m-1)/2} (1-r)^m (\frac{1-p}{p})^j \sum_{n=2j+1}^{k_j} \frac{m!}{(m-n)!}(1-r)^{-n}\frac{p^n}{n!}\binom{n}{j} r^n
\end{aligned}$$

The first step rearranges the order of addition. The final step introduces a series of variables which limit the number of terms in the inner sum. The inequality holds for all $k_j \leq m$. Notice that the k_j may vary for each term of the outer sum. Since $n \leq k_j \leq m$ we can use the following relation:

$$\frac{m!}{(m-n)!} \geq (m-k_j)^n \tag{8}$$

Letting $x_j = rp\frac{(m-k_j)}{(1-r)}$ we can simplify as follows:

$$\begin{aligned}
G(m,r,p) &\geq \sum_{j=0}^{(m-1)/2} (1-r)^m (\frac{1-p}{p})^j \sum_{n=2j+1}^{k_j} \binom{n}{j} \frac{m!}{(m-n)!}\frac{(1-r)^{-n}r^n p^n}{n!} \\
&\geq \sum_{j=0}^{(m-1)/2} (1-r)^m (\frac{1-p}{p})^j \sum_{n=2j+1}^{k_j} \binom{n}{j} (m-k_j)^n \frac{(1-r)^{-n}r^n p^n}{n!} \\
&= \sum_{j=0}^{(m-1)/2} (1-r)^m (\frac{1-p}{p})^j \sum_{n=2j+1}^{k_j} \binom{n}{j} \frac{x_j^n}{n!} \\
&\geq \sum_{j=0}^{g} (1-r)^m (\frac{1-p}{p})^j \sum_{n=2j+1}^{k_j} \binom{n}{j} \frac{x_j^n}{n!}
\end{aligned}$$

The last step introduces g and holds for all $g \leq (m-1)/2$. This is because in practice only the first few terms of the outer sum are significant. Thus for suitably chosen g, k_j this is a cheaply computable lower bound for G. A program to compute this to a high degree of accuracy has been implemented.

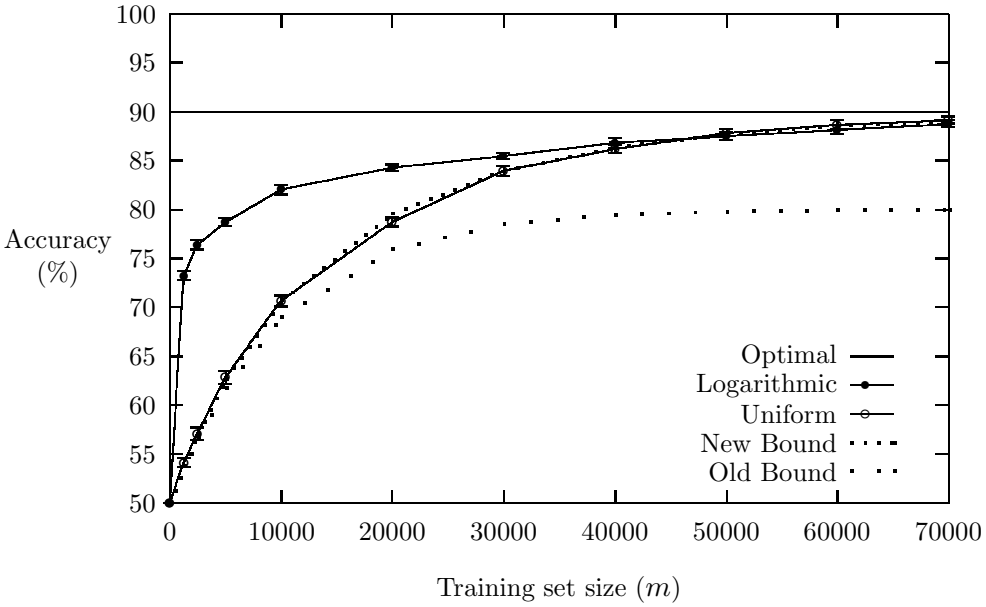

Figure 1: Simulation results and theoretical predictions for 10000 bins

4 Experiment

4.1 Skewed Bins

The assumption that bin probabilities are uniform is problematic. When bins are uniformly probable, the expected number of training instances in the same bin as a random test instance is $\frac{m}{|B|}$ ($= \sum_{b \in B} \Pr(b) \sum_{n=0}^{m} n \Pr(n \text{ training items fall into } b)$). But most distributions in language are highly skewed. Zipf's law states that word types are distributed logarithmically (the nth most frequent word has probability proportional to $\frac{1}{n}$). When this is true the expected number of training instances in the same bin as a random test instance is approximately $\frac{1.6m}{\log(0.56|B|)^2}$ ($\gg \frac{m}{|B|}$). Thus we can expect much more information to be available about typical test cases.

4.2 Simulations

Since the mathematics in section 3 cannot easily be generalised to different distributions, I have conducted several simulations in order to verify the mathematical results above and to explore the effect of using a skewed distribution of bins.

These simulations use a fixed number of bins (10,000), allocating m training instances to the bins according to either a uniform or logarithmic distribution. It then measures the correctness of the mode-based learner on 1000 randomly generated test instances to arrive at an observed correctness rate.[3]

This process (training and testing) is repeated 30 times for each run, with the mean being recorded as the observed accuracy. The standard deviation is used to estimate a 5% t-score confidence interval.

[3]The results were generated using an optimal value probability of $p = 0.9$ (thus the optimal accuracy rate is 90%). Simulations with other values of p did not differ qualitatively.

4.3 Results

Figure 1 shows five traces of accuracy as the volume of training data is varied. The lowest curve shows the old bound which can be achieved using the results in Lauer (1995), as represented by equation (5). The other dotted curve shows the expected accuracy predicted using equation (7) as approximated by the program described in section 3.2. The two further curves (with confidence interval bars) then show the results of simulations, using uniform and logarithmic bin distributions.

As can be seen, the new bound given in this paper is accurate for uniform bin probabilities. However, when the bins are logarithmically distributed learning converges significantly more quickly, as suggested by the reasoning about expected number of relevant training instances (see section 4.1). Perhaps surprisingly though, the logarithmic distribution appears to eventually fall behind the uniform one once there is plenty of data. This might be explained by the presence of very rare bins in the logarithmic distribution which thus take longer to learn. Both these observations are crucial to reasoning about data requirements for SLL.

5 Conclusion

If commercial NLP systems are to be developed from the current batch of research prototypes for SLL, then a predictive theory of the data requirements of such systems is necessary. In this paper I have explored the dependence of the expected accuracy of a simple statistical learner on the volume of training data. When the probability distribution of inputs is uniform, I have shown how to compute the expected accuracy, a result backed up by simulations. In particular, an average of four training instances per bin can be expected to yield an error rate only 50% worse than the optimal error rate.

When the distribution is non-uniform, simulations show that convergence can be much more rapid. Error rates only 50% worse than optimal result from only three training instances per bin. However, when data is abundant, non-uniform distributions result in higher error rates than the estimate produced by assuming uniformity.

6 Acknowledgements

I am grateful to Mark Johnson, without whom this work would not exist, and also to Robert Dale, Mark Dras, Mike Johnson and John Potter. Financial support is gratefully acknowledged from the Australian Government and the Microsoft Institute.

References

[1] Brent, Michael. 1993. From Grammar to Lexicon: Unsupervised Learning of Lexical Syntax. In *Computational Linguistics,* **Vol 19(2)**, pp243-62.

[2] Charniak, Eugene. 1993. *Statistical Language Learning.* MIT Press, Cambridge, MA.

[3] Dagan, I. and Itai, A. 1990. A Statistical Filter For Resolving Pronoun References. In *Proceedings of the Seventh Israeli Conference on Artificial Intelligence and Computer Vision*, Ramat Gan, Israel. pp125-35.

[4] de Haan, Pieter. 1992. Optimum Corpus Sample Size? In Leitner, Gerhard (ed.) *New Directions in English Language Corpora*. Mouton de Gruyter, Berlin.

[5] Hindle, D. and Rooth, M. 1993. Structural Ambiguity and Lexical Relations. In *Computational Linguistics* **Vol. 19(1)**, pp103-20.

[6] Lauer, Mark. 1995. How Much Is Enough? Data Requirements for Statistical NLP. In *Proceedings of the 2nd Conference of the Pacific Association for Computational Linguistics*, Brisbane, Australia. cmp-lg/9509001

[7] Lauer, M. and Dras, M. 1994. A Probabilistic Model of Compound Nouns. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, Armidale, NSW, Australia. World Scientific Press, pp474-81. cmp-lg/9409003

[8] Marcus, M., Marcinkiewicz, M. A. and Santorini, B. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. In *Computational Linguistics* **Vol 19(2)**, pp313-30.

[9] Yarowsky, David. 1992. Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In *Proceedings of COLING-92*, France, pp454-60.